ENGINEERING IN ADVANCED RESEARCHSCIENCE AND TECHNOLOGY

ISSN 2352-8648
Vol.03, Issue.01
September-2021
Pages: -356-364

IMPLEMENTATION OF HIGH SPEED QUATERNARY SIGNED DIGIT MULTIPLIER USING QSD ADDER

[1] *M. Tech., Dept of ECE, V.K.R., V.N.B., & A.G.K. College of Engineering. AP, India,* krishnavasa7@gmail.com
[2]*M. Tech., Dept of ECE, V.K.R., V.N.B., & A.G.K. College of Engineering. AP, India,* devakikoduru25@gmail.com

Abstract
This paper presents a high-speed Vedic multiplier based on the Urdhva Tiryagbhyam sutra of Vedic mathematics that incorporates a novel adder based on Quaternary Signed digit number system. Three operations are inherent in multiplication: partial products generation, partial products reduction and addition. Fast adder architecture therefore greatly enhances the speed of the overall process. Quaternary logic adder architecture is proposed that works on a hybrid of binary and quaternary number systems. A given binary string is first divided into quaternary digits of 2 bits each followed by parallel addition reducing the carry propagation delay. The design doesn't require a radix conversion module as the sum is directly generated in binary using the novel concept of an adjusting bit. The proposed multiplier design is compared with a Vedic multiplier based on multi voltage or multi value logic [MVL], Vedic Multiplier that incorporates a QSD adder with a conversion module for quaternary to binary conversion, Vedic multiplier that uses Carry Select Adder and a commonly used fast multiplication mechanism such as Booth multiplier. All these designs have been developed using Verilog HDL and synthesized by Synopsys Design Compiler. They have been realized using the open source NAN gate 15nm technology library. The proposal shows a maximum of 88.75% speed improvement with respect to Multi Value logic based 128x128 Vedic multiplier while the minimum is 17.47%.

Keywords: Quaternary Signed Digit adder [QSD]; Urdhva Tiryagbhyam; Vedic Mathematics

I. Introduction

One of the primary features that help us determine the computational power of a processor is the speed of its arithmetic unit. An important function of an arithmetic block is multiplication because, in most mathematical computations, it forms the bulk of the execution time. Thus, the development of a fast multiplier has been a key research area for a long time. Some of the important algorithms proposed for fast multiplication in literature are Array, Booth and Wallace multipliers [1]-[5]. Vedic Mathematics [6, 7] is a methodology of arithmetic rules that allows for more efficient implementations regarding speed. Multiplication in this methodology consists of three steps: generation of partial products, reduction of partial products, and finally carrypropagate addition. Multiplier design based on Vedic mathematics has many advantages as the

partial products and sums are generated in one step, which reduces the carry propagation from LSB to MSB. This feature helps in scaling the design for larger inputs without proportionally increasing the propagation delay as all smaller blocks of the design work concurrently. References [8], [9] and [11] compared Vedic Multiplier with other multiplier architectures namely Booth, Array and Wallace on the basis of delay and power consumption. Vedic multiplier showed improvements in both the parameters over other architectures. Thus, many implementations of multiplication algorithms based on Vedic sutras have been reported in literature [10]-[12]. Vedic multiplier schemes proposed in literature are based on Urdhva Tiryagbhyam and Nikhilam sutras of Vedic Mathematics. As Nikhilam sutra is only efficient for inputs that are close to the power of 10, in this paper a design to perform high-speed multiplication based on the Urdhva Tiryagbhyam sutra of Vedic Mathematics which is generalized method for all numbers, has been presented. The final step, carry-propagate addition, requires a fast adder scheme because it forms a part of the critical path. A variety of adder schemes have been proposed in literature to optimize the performance of Vedic multiplier [13]. Adder based on QSD shows an improvement in speed over other state of the art adders [14, 15]. Earlier implementations of QSD adder were based on Multi Voltage or Multi Value Logic (MVL) [16]. The difficulty in application of quaternary addition outside MVL (Multi Voltage logic) is that, the adder is only a small unit of the design whose outputs will needed to be converted back to binary for further processing. However, use of a conversion module undermines the advantages gained in speed by using QSD. In this paper, a novel implementation of an adder based on QSD is proposed, which reduces the carry propagation delay in the design by making use of carry free arithmetic. The proposed adder design works on a hybrid of binary and quaternary number systems wherein the sum is directly generated in binary using the concept of an adjusting bit, eliminating the conversion module. The design can be scaled to larger bit implementations such as 32, 64, 128 or more with minimal increase in propagation delay owing to the parallelism prevalent in the design. We have compared our design with a Vedic multiplier based on MVL logic that uses a ripple carry adder [16], Vedic Multiplier that incorporates a QSD adder and a conversion module for quaternary to binary conversion, Vedic multiplier that uses state of the art fast adder scheme such as Carry select adder [17] and a commonly used fast multiplication mechanism such as Booth multiplier [18], to prove the feasibility of our design across important comparison points.

II. BASIC TERMINOLOGY

A. Urdhva Tiryagbhyam (UT) Sutra The UT sutra is an ancient Vedic Mathematics sutra that can be used for multiplication of two numbers in any number system. It is based on “Vertical and Crosswise” multiplication. A 2x2 multiplier based on UT sutra is depicted in Fig. 1 and Fig. 2, where X and Y represent inputs while Z corresponds to output. Stepwise procedure is outlined below.

Step1: Vertical Multiplication: The least significant digits of the multiplicand and the multiplier are multiplied, as in (1).

$$Z0=X0.Y0 \quad (1)$$

Step2: Crosswise Multiplication and Addition: Z1, in (2), is obtained by cross multiplying X1 and Y0, and Y1 and X0 and subsequently adding the two products. In this stage a carry C1, as in (3), might be generated, that is propagated to the next step.

$$Z1 = (X0.Y1) \oplus (X1.Y0) \quad (2)$$

$$C1 = X0.X1.Y0.Y1 \quad (3)$$

Step3: Vertical Multiplication and Addition: The most significant digits of the multiplicand and the multiplier are multiplied, and the product is added with the carry of the previous step to obtain Z3 and Z2, as in (4) and (5) respectively.

$$Z2 = (X1.Y1) \oplus C1 \quad (4)$$

$$Z3 = X0.X1.Y0.Y1 \quad (5)$$

The final result is concatenation of Z3, Z2, Z1 and Z0. Fig. 1. Vertical and Crosswise multiplication The logic circuit for 2x2 UT multiplier is shown Fig. 2. Fig. 2. 2x2 UT multiplier.

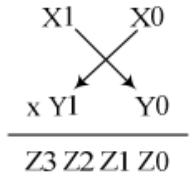


Fig. 1. Vertical and Crosswise multiplication

The logic circuit for 2x2 UT multiplier is shown Fig. 2.

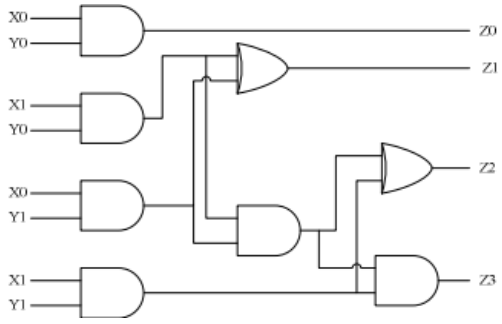


Fig. 2. 2x2 UT multiplier

B. Quaternary Signed Digit (QSD number system)

The QSD is a radix-4 number system that provides the benefit of faster arithmetic calculations over binary computation, as it eliminates rippling of carry during addition. Every number in QSD can be represented using digits from the set {-3,-2,-1, 0, 1, 2, 3}. Being a higher radix number system it utilizes less number of gates and hence saves on time and reduces circuit complexity. The stages involved in addition of two numbers in QSD are: Stage1: Generation of intermediate carry and sum: When two digits are added in QSD number system, the resulting sum ranges between -6 to +6. Numbers with magnitude higher than 3 are represented by multiple digits with least significant digit representing sum and the next digit corresponds to carry. Also, every number in QSD can have multiple representations [14, 15]. The representation is chosen such that the magnitude of sum digit is 2 or less than 2 and the magnitude of carry digit is 1 or less than 1, the reason for which is explained in the next stage. Stage2: The intermediate sum and carry have a limit fixed on their magnitude because this allows carry free addition in the second step. The result can be obtained directly by adding the sum digit with the carry of the lower significant digit [14, 15].

III. PROPOSED DESIGN

A. 4x4 Multiplier

Block diagram of a 4x4 multiplier is shown in Fig. 3. In this multiplier, four 2x2 multipliers are arranged systematically. Each multiplier accepts four input bits; two bits from multiplicand and other two bits from multiplier. Addition of partial products is done using two four bit Quaternary adders, a two-bit adder and a half adder. The final result is obtained by concatenating the least significant two bits of the first multiplier, four sum bits of the second four-bit

Quaternary adder and the sum bits of two-bit adder.

Table I shows all intermediate and final results involved in the multiplication process of two binary numbers, A = (1111)2 and B = (1001)2. The data flow in the proposed 4x4 multiplier is given below: 1) A[1:0] and B[1:0], A[3:2] and B[1:0], A[1:0] and B[3:2], and A[3:2] and B[3:2] are multiplied by 2x2 Vedic multipliers, giving output D0[3:0], D1[3:0], D2[3:0] and D3[3:0] respectively.

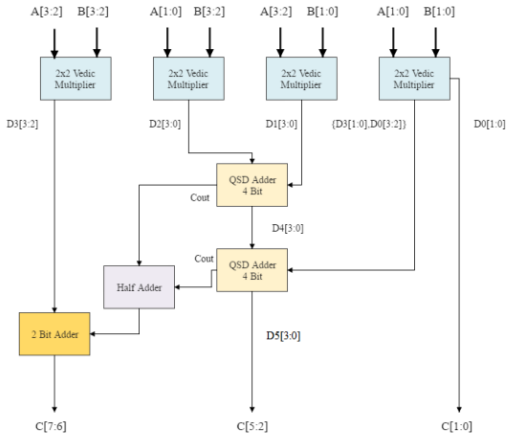


Fig. 3. Proposed 4x4 Multiplier

TABLE I. MULTIPLICATION RESULT OF TWO 4 BIT BINARY NUMBERS USING THE PROPOSED DESIGN

	Binary equivalent	Decimal equivalent	Explanation
A	$(1111)_2$	15	Input 1
B	$(1001)_2$	9	Input 2
D0	$(0011)_2$	3	Output of 2x2 Vedic Multiplier 1
D1	$(0011)_2$	3	Output of 2x2 Vedic Multiplier 2
D2	$(0110)_2$	6	Output of 2x2 Vedic Multiplier 3
D3	$(0110)_2$	6	Output of 2x2 Vedic Multiplier 4
D4	$(01001)_2$	9	Output of 4 bit QSD adder 1 (D1+D2)
D5	$(10001)_2$	17	Output of 4 bit QSD adder 2 (D4 +{D3[1:0],D0[3:2]})
C[1:0]	$(11)_2$	3	D0[1:0]
C[5:2]	$(0001)_2$	1	D5[3:0]
C[7:6]	$(10)_2$	2	Output of 2 Bit Adder (D3[3:2]+D4[4]+D5[4])
C[7:0]	$(10000111)_2$	135	Final Result

2) D1 [3:0] and D2[3:0] are added by the proposed 4 bit QSD adder, giving D4[3:0] and a carry out as the outputs. 3) D4[3:0] and {D3[1:0], D0[3:2]} are added by the second 4 bit QSD adder, giving D5[3:0] and a carry out as the outputs. 4) The half adder is used to add the carry outs of the QSD adders. The output obtained is fed to the 2 Bit Adder along with D3[3:2]. 5) The result, C, in binary is obtained by concatenation of output of 2 Bit Adder, D5[3:0] and D0[1:0]. The proposed design can be extended to multiply both negative and positive integers by an addition of a sign bit in both inputs. An XOR logic can then be used to compute the sign bit of the final output. The multiplication of the magnitudes will proceed simultaneously in a similar manner to the example described above.

B. 32x32 multiplier

The 4x4 multiplier design can be scaled to multiply larger numbers as shown in Fig. 4, where the design is scaled up for a 32 bit multiplier.

C. Proposed adder design based on QSD

In this paper, a novel idea of an adder, based on QSD (Quaternary Signed Digit) is proposed. The algorithm for the proposed adder uses a hybrid of quaternary and binary number systems. The outputs from smaller multipliers are obtained as binary strings. Inside the addition module, this string is broken into quaternary digits of two bits each.

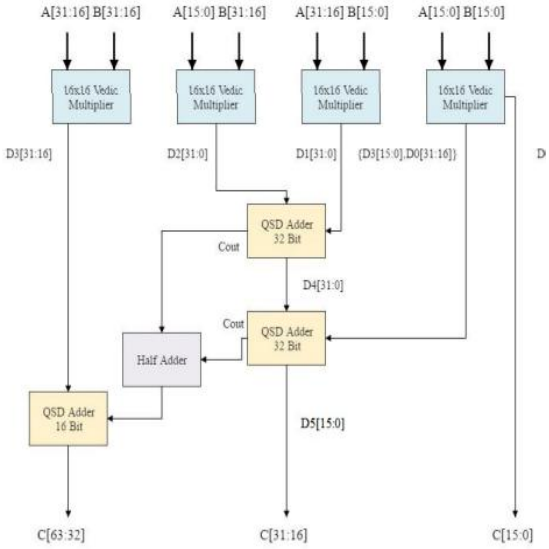


Fig. 4. Proposed 32x32 Multiplier

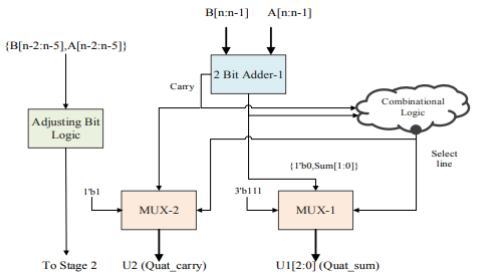
TABLE II. CONVERSION OF A QUATERNARY NUMBER TO BINARY NUMBER SYSTEM

Quaternary number A	$2\,1 \rightarrow 010\_001$	Quaternary number B	$2\,\bar{1} \rightarrow 010\_111$
Binary equivalent of A	1001	Incorrect Binary equivalent of B	1011
Decimal equivalent of A	9	Incorrect Decimal equivalent of B	11

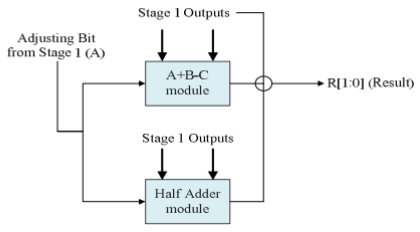
Addition using QSD allows us to reduce the carry propagation delay by making use of carry free arithmetic i.e. the carry doesn't ripple past the subsequent quaternary digit. Especially for higher bit input strings this method is extremely efficient. The difficulty in application of quaternary addition outside MVL (Multi Voltage logic) is that the least significant 2 bits of the binary representation of the quaternary digits can't be directly concatenated to form an output binary string for every case as depicted in Table II. Each string would have to be read individually and a conversion module that converts quaternary to binary would have to be employed. To overcome this limitation, the concept of an adjusting bit has been introduced.

The Intermediate sum lies in the range [0, 6], as the operands are unsigned numbers. From [16], for quaternary addition to be carry free beyond the first stage, the intermediate sum can't be greater than 2. To ensure this stipulation holds true, the (1)4 representation of 3 needs to be chosen while adding. However, this represents a blocking case when converting the final output string back into binary as it prohibits us from simply concatenating the lower two bits of quaternary output strings to get the binary equivalent.

For addition of unsigned numbers, if the (03)4 representation would have been used, direct concatenation of results could have been possible. But, then that wouldn't have always been carry free after the initial stage. Thus, the concept of an adjusting bit has been devised to solve the dilemma of which representation of 3 to use, such that both carry free addition and concatenation of output string bits to get the final output can be realized in the same design. The solution to the problem described above, is that the (03)4 representation of 3 is required to be taken instead of the (1)4 representation in some cases. But, determining when such a change is required before proceeding with the addition will increase the delay of the design and be counter-productive. Thus, the (1)4 representation of 3 is always selected in stage 1, to satisfy necessary conditions for carry free arithmetic. While necessary adjustments are made in stage 2 if (03)4 representation was to be taken, the need for such an adjustment is determined via an adjusting bit.



(a) Stage 1



(b) Stage 2

Fig. 5. Proposed Adder

The proposed adder works in two stages, as shown in Fig. 5. 1) In the first stage, as

in Fig. 5(a), every individual digit at the same position in the quaternary representation of two n-bit numbers A and B is added using a 2 Bit Adder to generate a sum. This sum lies in the range [0, 6]. From the sum obtained from the adder, the intermediate sum and intermediate carry for the next stage are calculated in parallel using 2x1 multiplexers. The logic for the selection of the representation of sum and carry has been explained in [16].

The adjusting bit is also computed in parallel with the addition process. The input to the adjusting bit calculation block for every quaternary digit addition are the previous two quaternary digits of A and B signified by [n-2: n-5]. 2) Second stage has two modules as shown in Fig. 5(b). One is a one-bit module that performs the computation (A+BC). In this case A would be LSB of intermediate sum, B would be carry from the previous quaternary digit addition and C would be the adjusting bit. The other module will be a half adder which will add the carry from the (A+B-C) module and the bit to the left of the least significant bit of the intermediate sum. As for the final concatenation, the sign bit would not be used owing to the adjustments proposed in the design.

IV.SIMULATION RESULTS

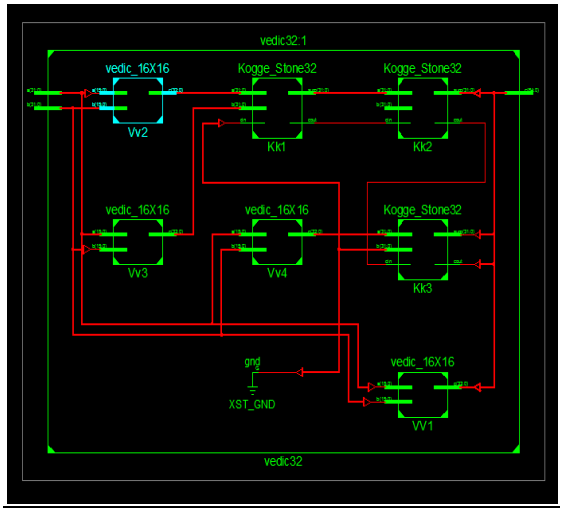


Figure 6 : RTL Schematic

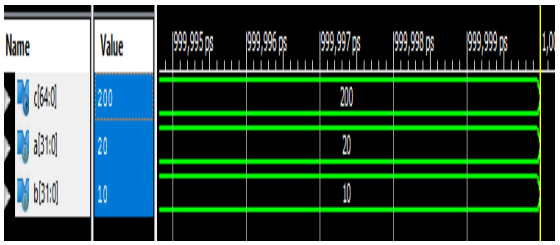


Figure 7: simulation outcome

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	2443	303600	0%
Number of fully used LUT-FF pairs	0	2443	0%
Number of bonded IOBs	133	700	19%

Figure 8: Design summary

```
LUT5:I1->O            2    0.043   0.500  VV11/Kk3/K1/GC3/G (VV11/Kk3/K1/s<3>)
LUT6:I3->O            2    0.043   0.355  VV11/Kk3/K1/GC7/G3_SW0 (N288)
LUT5:I4->O            1    0.043   0.350  VV11/Kk3/K1/GC7/G3 (VV11/Kk3/K1/GC7/G2)
LUT6:I5->O            1    0.043   0.405  VV11/Kk3/K1/GC16/G4_SW0 (N256)
LUT6:I4->O            4    0.043   0.620  VV11/Kk3/K1/GC16/G4 (VV11/Kk3/cout1)
LUT6:I1->O            4    0.043   0.539  VV11/Kk3/K2/GC1/G1 (VV11/Kk3/K2/q<1>)
LUT6:I2->O            2    0.043   0.608  VV11/Kk3/K2/GC6/G11 (VV11/Kk3/K2/GC6/G1)
LUT5:I0->O            3    0.043   0.507  VV11/Kk3/K2/GC8/G11 (VV11/Kk3/K2/GC8/G1)
LUT6:I3->O            2    0.043   0.608  VV11/Kk3/K2/GC10/G1 (VV11/Kk3/K2/v<10>)
LUT6:I1->O            1    0.043   0.405  VV11/Kk3/K2/GC14/G1 (VV11/Kk3/K2/v<14>)
LUT6:I4->O            1    0.043   0.339  Mmux_out321 (out_63_OBUF)
OBUF:I->O                 0.000          out_63_OBUF (out<63>)
----------------------------------------
Total                    23.886ns (2.860ns logic, 21.026ns route)
                                  (12.0% logic, 88.0% route)
```

Figure 9: Time summary

V. CONCLUSION

It can be concluded that the design when scaled to higher bits only shows a marginal rise in delay due to its core strengths. Firstly, the parallelism involved in its partial product generation. Secondly, reduction of carry propagation delay in the novel adder it incorporates. Due to the use of QSD, the design is able to incorporate carry free arithmetic while eliminating radix conversion module speed overhead by integrating concept of adjusting bit logic in its architecture. The proposed design showed an increase in implementation area over some designs due to increased parallelism even in finer nuances of the architecture. The proposed design is targeted towards digital systems requiring high throughput and low latency at the cost of area overhead. For example, in a DSP system, operations such as Fast

Fourier Transform, Convolution, Filtering and Discrete Wavelet transform etc. Multipliers play a key role in determining the speed of the system. Similarly, this architecture would be a good candidate to be implemented as a large part of systems like DCT, Central Processing Unit (CPU), MAC (Multiply and Accumulate) Unit, Image Processors where high-speed multiplications are critical to the performance of the system. It can also be observed that despite the objective of decreasing the delay, the proposed design performs better than most designs compared in terms of power for lower input bit sizes [16 and 32 bit]. Although it consumes more power than other designs higher input bit sizes [64 and 128 bit], it is justifiable when factored in with advantages gained in speed for higher input bits.

REFERENCES

[1]. GarimaRawat, KhyatiRathore, SiddharthGoyal,Shefali Kala and Poornima Mittal, (2015). "Design andAnalysis of ALU: Vedic Mathematics". IEEE Int. Conf. on Computing, Communication and Automation (ICCCA2015), pp. 1372-1376.

[2]. Rahul Nimje and ShardaMungale, (2014). "Design of arithmetic unit for high-speed performance using Vedic mathematics". International Journal of Engineering Research and Applications, pp. 26-31.

[3]. Poornima M, Shivaraj Kumar Patil, Shivukumar, Shridhar K P and Sanjay H, (2013). "Implementation of multiplier using Vedic algorithm".International Journal of Innovative Technology and Exploring Engineering, Vol. 2, No. 6.

[4]. M. Sowmiya, R. Nirmal Kumar, S.Valarmathy and S. Karthick, (2013). "Design of Efficient Vedic Multiplier by the analysis of Adders". International Journal of Emerging Technology and Advanced Engineering, Vol. 3, No.1.

[5]. PushpalataVerma and K. K. Mehta, (2012). "Implementation of an Efficient Multiplier based on Vedic Mathematics Using EDA Tool". International Journal of Engineering and Advance Technology, Vol.1, No. 5.

[6]. Abhishek Gupta,UtsavMalviya and VinodKapse, (2012). "A novel approach to design high-speed arithmetic logic unit based on ancient Vedic multiplication technique". International Journal of Modern Engineering Research, Vol. 2, No. 4.

[7]. SuchitaKamble and N. N. Mhala, (2012). "VHDL implementation of 8-bit ALU".IOSR Journal of Electronics and Communication Engineering, Vol. 1, No. 1.

[8]. PushpalataVerma, (2012). "Design of 4x4 bit Vedic Multiplier using EDA Tool". International Journal of Computer Applications, Vol. 48, No. 20.

[9]. AniruddhaKanhe, Shishir Kumar Das and Ankit Kumar Singh, (2012). "Design and Implementation of Low Power Multiplier Using Vedic Multiplication Technique".International Journal of Computer Science and Communication (IJCSC), Vol. 3, No. 1, pp. 131-132.

[10]. UmeshAkare, T.V. More and R.S. Lonkar, (2012)."Performance Evaluation and Synthesis of Vedic Multiplier ". National Conference on Innovative Paradigms in Engineering & Technology (NCIPET-2012), Proceedings published by International Journal of Computer

Applications (IJCA), pp. 20-23.
[11]. Anvesh Kumar and Ashish Raman, (2010). “Low Power ALU Design by Ancient Mathematics”. IEEE, 978-1-4244-5586-7/10
[12]. Parth Mehta and DhanashriGawali, (2009). “Conventional versus Vedic mathematics method for hardware implementation of a multiplier”.International Conference on Advances in Computing, Control, and Telecommunication Technologies, pp. 640-642.
[13]. Ramalatha, M.Dayalan, K D Dharani, P Priya and S Deoborah, (2009). “High speed energy efficient ALU Design using Vedic Multiplication Techniques”. IEEE Int. Conf. on Advances in Computational Tools for Engineering Applications (ACTEA-2009), pp. 600-603.
[14]. Honey DurgaTiwari, GanzorigGankhuyag, Chan Mo Kim and Yong BeomCho, (2008). “Multiplier design based on Ancient Vedic Mathematics”.IEEE, 978-1-4244- 2599-0/08/$25.00 © 2008.
[15]. Jagadguru Swami Sri Bharati Krishna TirthjiMaharaja, (1986). Vedic Mathematics.MotilalBanarsidas, Varanasi, India.

Vasa Sai Krishna is a student of V.K.R., V.N.B., & A.G.K. College of Engineering,Gudivada.He is studying M.tech[ECE] and also received B.tech degree from Andhra University.

Koduru Devaki Devi is a Associate professor in V.K.R., V.N.B., & A.G.K. College of Engineering,Gudivada.She received Master Degree from different Universities.Having 11+ years of Experience as a faculty and Guide for Different Domains.